Deterministic Multi-Channel Information Exchange

Stephan Holzer
Distributed Computing Group
ETH Zurich, Switzerland
stholzer@ethz.ch

Thomas Locher
ABB Corporate Research
Baden-Dättwil, Switzerland
thomas.locher@ch.abb.com

Yvonne-Anne Pignolet
ABB Corporate Research
Baden-Dättwil, Switzerland
yvonne-anne.pignolet@ch.abb.com

Roger Wattenhofer
Distributed Computing Group
ETH Zurich, Switzerland
wattenhofer@ethz.ch

ABSTRACT

In this paper, we study the information exchange problem on a set of multiple access channels: k arbitrary nodes have information they want to distribute to the entire network via a shared medium partitioned into channels. We present algorithms and lower bounds on the time and channel complexity for disseminating these k information items in a single-hop network of n nodes. More precisely, we devise a deterministic algorithm running in asymptotically optimal time $\mathcal{O}(k)$ using $\mathcal{O}(n^{\log(k)/k})$ channels if $k \leq \frac{1}{6}\log n$ and $\mathcal{O}(\log^{1+\rho}(n/k))$ channels otherwise, where $\rho > 0$ is an arbitrarily small constant. In addition, we show that $\Omega(n^{\Omega(1/k)} + \log_k n)$ channels are necessary to achieve this time complexity.

Categories and Subject Descriptors

F.2.3. [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity—*Tradeoffs among Complexity Measures*

General Terms

Algorithms, Theory

Keywords

Information Dissemination, Wireless Networks, Single-Hop, Multi-Channel, No Collision Detection.

1. INTRODUCTION

A fundamental problem of many communication systems that rely on a shared communication medium, e.g., wireless and bus networks, is (co-channel) interference, which occurs when more than one network entity tries to transmit a message over the same communication channel at the same time. Such simultaneous (or interleaved) transmissions of two or more messages over the same channel are commonly referred to as *collisions*. Typically, a collision distorts all transmitted messages significantly, which entails that none of the messages can be decoded successfully at the receivers. Hence, there is a need for mechanisms scheduling the message transmissions appropriately in order to enable an efficient exchange of messages over the communication medium. There are various techniques to address or simplify this basic scheduling problem: By introducing a notion of time, the network entities can transmit in synchronized *time slots*, which reduces the potential for collisions. Another common trick is to use randomization, as in, e.g., the Aloha protocol. If the network entities further have the ability to detect collisions, which allows the entities to learn that other entities strive to transmit as well, back-off mechanisms can be applied to ensure an eventual transmission of all messages.

Moreover, in various communication systems several non-conflicting communication channels are available, which can be leveraged to disseminate information. While there is a lot of work on scheduling message transmissions for various models of communication channels, surprisingly little is known about the benefits and limits of using multiple channels for the purpose of information dissemination. This is the focus of this paper, which addresses the question of how many communication channels are required in order to solve an information exchange problem as quickly as possible. More generally, we study the power of having additional channels at one's disposal when trying to disseminate information. We believe that this is an important missing piece in the study of communication over shared channels. Before giving a more formal definition of the considered information exchange problem, we present the communication model used throughout this paper.

1.1 Model

In this paper, we consider a simple network topology, the complete (single-hop) communication network in which every node can communicate with every other node. There are n nodes in total, each with a given, unique identifier in the range $[n] := \{1, \ldots, n\}$ (when using an initialization algorithm that assigns identifiers to nodes, e.g., [17, 18], this assumption can be dropped). We assume that multiple channels are available for communication and that local computations require zero time (since we focus on communication complexity). Additionally, we make the simplifying assumption that time is divided into synchronized time slots, i.e., we study slotted protocols: In any time slot, each node v may choose a channel i and perform exactly one of two operations, `send`, which means that v *broadcasts* a message

SPAA'12, June 25–27, 2012, Pittsburgh, Pennsylvania, USA.

on channel i or `receive`, in which case v *listens* on channel i.[1] A transmission is *successful* if and only if exactly one node transmits its message on a given channel in a specific time slot. A node listening on a particular channel i only receives a message in a given time slot if there is a successful transmission on this channel. Messages are of bounded size, i.e., we assume that each message can only contain one information item (e.g., a node identifier). We further assume that there is no *collision detection*, i.e., if a node v does not receive a message when listening on a channel i, node v cannot determine whether there was a collision or no message was sent. This is a reasonable assumption as, e.g., simple wireless devices often do not have a reliable collision detection mechanism. Moreover, solutions in this model can be applied in settings with collision detection but not vice versa. We study the following problem in this communication model without collision detection.

Definition 1. (Information Exchange Problem.) There is an arbitrary subset of $k \leq n$ nodes (called *reporter nodes* or simply *reporters*) where each of the k nodes is given a distinct piece of information. This subset is determined by an adversary before the first time slot. The objective is to disseminate these k information items to every node in the network. The subset of reporters is not known initially. The number n of nodes and the number k of reporters may or may not be known.

This problem lies between two fundamental information dissemination problems: broadcasting (one-to-all communication) and gossiping (total information exchange). In other words, we generalize the *Information Exchange Problem* [12] (also known as *k-Selection* [15] and *Many-to-All Communication* [8]) for networks with several communication channels. In order to measure the quality of a solution to the *Information Exchange Problem*, we must define adequate complexity measures. Clearly, it takes a certain number of time slots to distribute all information items. As mentioned before, the goal is to disseminate all information items as quickly as possible. Therefore, the primary objective pursued in this paper is to find an algorithm $\mathcal{A}$ with an optimal *time complexity*, which is defined as the maximum number of synchronous time slots that $\mathcal{A}$ requires to disseminate all k items for a worst-case selection of reporters. Since only one information item can be transmitted in any message, i.e., items cannot be bundled, and each node can only listen on one channel per time slot, it follows that the time complexity of any algorithm is at least $\Omega(k)$. The key question thus becomes how many channels does an algorithm for the *Information Exchange Problem* require in order to achieve an asymptotically optimal time complexity of $\Theta(k)$? Chlebus and Kowalski [7] prove that it is not possible to disseminate all information items in time $\mathcal{O}(k)$ with only one communication channel by giving a lower bound of $\Omega(k + \log n)$. If more channels are available, the lower bound $\Omega(k)$ can be matched using *randomized* algorithms [14]. However, these algorithms need a large number of channels and there is a (small) probability that these algorithms fail.

1.2 Contributions

In this paper, we propose *deterministic* algorithms for the *Information Exchange Problem* when n and k are known. In particular, we introduce two algorithms both exhibiting an asymptotically optimal time complexity of $\Theta(k)$, which are appropriate for different values of k, and give bounds on the number of channels that each algorithm requires for a given interval of k. The first algorithm, called Algorithm $\mathcal{A}_S$, is useful for small values of k, that is for $k \leq \frac{1}{6}\log n$,[2] and requires $\mathcal{O}(n^{\log(k)/k})$ channels. For larger values of k we apply Algorithm $\mathcal{A}_L$ using $\mathcal{O}(\log^{1+\rho}(n))$ channels for some constant $\rho > 0$ when $k \in (\frac{1}{6}\log n, \log(n) \cdot \log\log(n))$ and $\mathcal{O}(\log(n/k))$ channels for larger k up to $n - 2\lceil \log n \rceil$. Note that for $k > n - 2\lceil \log n \rceil$ we can simply iterate over all nodes to find the reporters in time $\mathcal{O}(k) = \mathcal{O}(n)$, therefore we ignore this case in the remainder of this paper.

We complement these results with a lower bound on the number of channels that any deterministic algorithm needs in order to achieve an optimal time complexity: Any deterministic algorithm with a time complexity of $\mathcal{O}(k)$ must use at least $\Omega(n^{\Omega(1/k)} + \log_k n)$ channels (Theorem 5). The following table summarizes these results.

Range of k	$\left[1, \frac{1}{6}\log n\right]$	$\left(\frac{1}{6}\log n, \log(n)\log\log n\right)$
Algorithm	$\mathcal{A}_S$ (Thm. 3)	$\mathcal{A}_L$ (Thm. 4)
Channels	$\mathcal{O}\left(n^{\log(k)/k}\right)$	$\mathcal{O}\left(\log^{1+\rho}(n)\right)$
Lower Bound	$\Omega\left(n^{\Omega(1/k)}\right)$	$\Omega\left(\log(n)/\log\log(n)\right)$

R.	$[\log(n) \cdot \log\log(n), n - 2\lceil \log n \rceil)$	$[n - 2\lceil \log n \rceil, n]$
A.	$\mathcal{A}_L$ (Thm. 4)	Text above
C.	$\mathcal{O}(\log(n/k))$	1
L.	$\Omega(\log_k n)$	1

We derive the lower bound on the number of channels by first proving a lower bound on the time complexity when the (maximum) number of channels c is given. The lower bounds for a given number c of channels are of interest since in reality the number of available channels is often limited to a number c and does not grow with n or k. If c channels are available, the lower bound on the time complexity of deterministic algorithms is $\Omega(\log_c(n/k) + k)$ (see proof of Theorem 5 combined with $\Omega(k)$). This lower bound holds even in a less restrictive model where nodes can detect collisions and listen on all channels simultaneously. In light of this, it is surprising that the proposed algorithms are almost able to match the given lower bounds for certain values of k and n.

1.3 Related Work

Several papers study the information exchange problem for single-channel and multi-channel networks without collision detection. Kowalski [15] proves the existence of an oblivious deterministic algorithm without collision detection that distributes k information items on a single channel in time $\mathcal{O}(k\log(n/k))$ based on selectors as well as a matching lower bound. Moreover, he presents an explicit polynomial-time construction with time complexity $\mathcal{O}(k \operatorname{polylog} n)$ to solve this problem deterministically. Later these results have been improved and extended by Chlebus et al. [7] to multi-hop networks and the authors provide bounds for centralized and distributed algorithms. In contrast to our assumptions, they assume that all k information items fit into one message. When restricted to single hop networks, they present

[1] Naturally, a node may also choose not to perform any operation in a given time slot.

[2] Note that the base of the logarithm is 2 throughout the paper.

a randomized algorithm for one channel that disseminates all information items in time $\mathcal{O}(\log(k) \cdot (\log^2 n + k))$ whp$_k$, i.e., with probability at least $1 - 1/k^\lambda$, where $\lambda \geq 1$ is a parameter in the algorithm or in the analysis.

Kushilevitz and Mansour [16] proved a lower bound of $\Omega(k + \log n)$ on the expected time of randomized algorithms. The average time complexity in directed networks is addressed by Chlebus et al. [8] who present an upper and a lower bound of $\mathcal{O}(\min\{k \log(n/k), n \log n\})$ and $\Omega(k/\log n + \log n)$, respectively. Moreover, they devised a protocol for the case when information items have to be delivered separately as in our model within time $\mathcal{O}(k \log(n/k) \cdot \log n)$ and a lower bound of $\Omega(k \log n)$.

Recently, Fernandez et al. [1] presented a randomized algorithm for single-channel, single-hop networks that works without information on the number of contenders and of the size of the network in time $\mathcal{O}(k)$ whp$_k$. The authors of [14] showed that better bounds can be achieved by exploiting the availability of multiple channels: the dissemination problem can be solved with an asymptotically optimal time complexity of $\Theta(k)$. However, the randomized algorithms provided in their paper require $\sqrt{n}$ channels for $k < \sqrt{\log n}$ and $n^{\log(k)/k}$ channels for $\sqrt{\log n} < k < \log n$. Moreover, their deterministic algorithm uses n channels.

The information exchange problem in networks suffering from adversarial interference has been studied in [10, 11] where n nodes inform each other about $n - t$ values and an adversary can disturb communication on t channels by jamming.

In a recent paper by Gilbert and Kowalski [12] upper and lower bounds are given for the information exchange problem in single-channel networks where some of the nodes exhibit Byzantine behavior.

The closely related problems of consensus and mutual exclusion have lately been studied in [3, 9] for single-channel networks with and without a global clock, collision detection, and knowledge of the number of nodes in the network. Some parts of our algorithms are inspired by the algorithms presented in [6]. In this paper, Chlebus and Kowalski propose algorithms based on lossless expander graphs for the *renaming problem* [2]. In the renaming problem, each of n processes initially has a unique identifier in the range $[n] := \{1, \ldots, n\}$. The goal is to assign new unique names[3] from a smaller range to a subset of k processes using r shared registers. The algorithm must be correct for every selection of k processes. The renaming problem has been studied in a variety of communication models, mainly in shared memory and message passing models (see [5] for a recent survey). The time complexity of renaming algorithms depends on the communication model, the (un)known parameters, the number of reporters relative to the network size, and the range of the output names.

We adapt the compete operation of [6] for MAC models and introduce a new class of bipartite graphs as a base for our renaming algorithms. The special nature of the communication medium and the fact that no external devices (such as registers) can be used requires new ideas. We prove the existence of graphs with different properties leading to better results for our model. To the best of our knowledge, this paper is the first to provide renaming algorithms for MAC.

2. BUILDING BLOCKS

While the algorithms introduced in the subsequent section are based on different techniques, they still share certain basic algorithmic ideas, which are discussed in this section. Note that throughout the paper we assume that $k \geq 2$ as the information exchange problem is trivial for $k \leq 1$.

2.1 Matching Graphs

A core concept used in our algorithms is a special class of bipartite expander graphs $G = (V \cup W, E)$, where the edges in E connect the nodes in the two disjoint node sets V and W, which we refer to as *matching graphs.* In these graphs each node is provided with a fixed ordering of its incident edges. Using this order, a *weak unique-neighbor* property is satisfied in matching graphs: for any subset $X \subseteq V$ of a certain maximum size and an arbitrarily small but fixed constant parameter $\varepsilon \in (0, 1)$, there is an edge index i such that at least $\lceil \varepsilon |X| \rceil$ nodes in W are adjacent to exactly one (and thus a unique) neighbor $v \in X$ when we consider the subgraph G_i induced by the i^{th} edges of each node $v \in X$. Therefore, a matching between $\lceil \varepsilon |X| \rceil$ nodes in X and nodes in W can be found by iterating over the edges according to the fixed edge order of the nodes in X. These graphs have certain expansion properties that are implied by their unique-neighbor property. Note that matching graphs are inspired by lossless expanders (see, e.g., [4]) used in the context of asynchronous exclusive selection [6]. While lossless expanders are well suited for asynchronous exclusive selection, the less restrictive matching graphs yield better results in our wireless setting.[4] Formally, matching graphs are defined as follows.

Definition 2. ((K, Δ, ε)-matching Graphs.) Let $G = (V \cup W, E)$ be a bipartite graph, where V and W are the disjoint node sets and $E \subseteq V \times W$ is the edge set. For each $v \in V$, there is an edge ordering and $\Gamma(v, i)$ denotes the i^{th} neighbor of v. G is a (K, Δ, ε)-*matching* graph if each $v \in V$ has Δ neighbors, and for each subset $X \subseteq V$ of size at most K, there is an index $1 \leq i \leq \Delta$ such that at least $\lceil \varepsilon |X| \rceil$ nodes in X have a unique i-neighbor. For each node $v \in X$ and index i, node $\Gamma(v, i)$ is a unique i-neighbor if $\Gamma(v, i) \neq \Gamma(w, i)$ for all $w \in X \setminus \{v\}$.

We would like to have a matching graph with a small node set W and a small degree Δ while keeping ϵ as large as possible. It is not hard to see that the minimum cardinality of W depends not only on Δ, K, and ε, but also on the size of V. Given these parameters, we prove that matching graphs exist if the following restriction on the minimum size of the node set W holds.

THEOREM 1. *For any $\varepsilon \in (0, 1)$, $\alpha > \frac{2}{1-\varepsilon}$, $K \geq 2$, and $\Delta \geq 1$, a (K, Δ, ε)-matching graph $G = (V \cup W, E)$ exists if the following two conditions are satisfied:*

$$|W| \geq |V|^{\frac{\alpha}{\Delta}} \quad (1)$$

$$|W| \geq e^{\frac{(1+\varepsilon)\alpha}{(1-\varepsilon)\alpha - 2}} \left(\frac{1+\varepsilon}{2} \right)^{\frac{(1-\varepsilon)\alpha}{(1-\varepsilon)\alpha - 2}} \cdot K^{\frac{(2-\varepsilon)\alpha}{(1-\varepsilon)\alpha - 2}} \quad (2)$$

PROOF. We use the probabilistic method to prove this statement. Specifically, we show that letting each node $v \in V$ choose Δ neighbors $\Gamma(v, 1), \Gamma(v, 2), \ldots, \Gamma(v, \Delta)$ in W uniformly at random results in a (K, Δ, ε)-matching graph

[3] We refer to the identifiers in the old namespace as "identifiers" while the identifiers in the new namespace are simply called "names".

[4] All lossless expanders are also matching graphs.

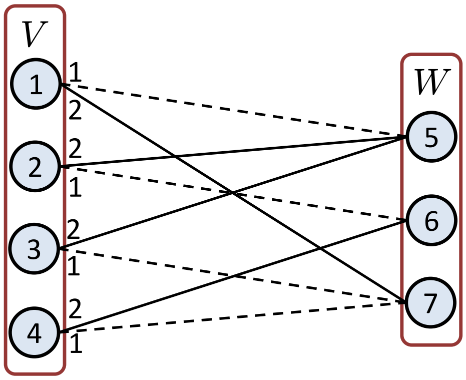

Figure 1: An example of a $(4, 2, 1/2)$-matching graph with $|V| = 4$ and $|W| = 3$ is shown. The index numbers on the edges define the order of the neighbors. Edges with index 1 are dashed, edges with index 2 are full lines. If we consider, e.g., the set $X = \{1, 3, 4\}$, the edges with index 1 do not provide a sufficiently large matching since two out of the three nodes share the same 1-neighbor (nodes 3 and 4). However, index 2 delivers a sufficiently large matching since all nodes (and thus at least half of the nodes in X) have a unique 2-neighbor. For $X = \{2, 3\}$ index 1 can be used, while index 2 cannot. For $X = \{1, 2\}$ both indices can be chosen. For any other subset $X \subseteq V$, there is always at least one index that works.

with positive probability if Condition (1) and Condition (2) are satisfied.

Given such a randomly constructed graph, consider any subset $X \subset V$ of cardinality $x \leq K$. Let the random variable N_i denote the number of nodes in W that are neighbors of the nodes in X if we only consider the i^{th} edge of each node $v \in X$. Formally, $N_i = |\{w \in W \mid \exists v \in X : \Gamma(v, i) = w\}|$. We now prove that the probability that N_i is at least $\lceil \frac{1+\varepsilon}{2} x \rceil$ is large. For this purpose, we need the following inequality:

$$\begin{aligned}
&|W|^{\left(\frac{1-\varepsilon}{2} - \frac{1}{\alpha}\right)x} \\
&\overset{(2)}{\geq} \left(e^{\frac{(1+\varepsilon)\alpha}{(1-\varepsilon)\alpha - 2}} \left(\frac{1+\varepsilon}{2}\right)^{\frac{(1-\varepsilon)\alpha}{(1-\varepsilon)\alpha-2}} \cdot K^{\frac{(2-\varepsilon)\alpha}{(1-\varepsilon)\alpha-2}} \right)^{\left(\frac{1-\varepsilon}{2} - \frac{1}{\alpha}\right)x} \\
&= e^{\frac{1+\varepsilon}{2}x} \left(\frac{1+\varepsilon}{2}\right)^{\frac{1-\varepsilon}{2}x} \cdot K^{\frac{1-\varepsilon}{2}x + \frac{x}{2}} \\
&\overset{K \geq x \geq 2}{\geq} e^{\frac{1+\varepsilon}{2}x} \left(\frac{1+\varepsilon}{2}\right)^{\frac{1-\varepsilon}{2}x} \cdot x^{\frac{1-\varepsilon}{2}x+1} \\
&> e^{\frac{1+\varepsilon}{2}x} \left(\frac{1+\varepsilon}{2}x\right)^{\frac{1-\varepsilon}{2}x+1} . \qquad (3)
\end{aligned}$$

Since there are $\binom{|W|}{j}$ ways to choose j nodes in W and there are at most j^x ways to choose neighbors for the nodes in X in such a way that all j nodes are chosen at least once, we get that $\mathbb{P}[N_i = j] \leq \binom{|W|}{j} \frac{j^x}{|W|^x}$. The probability that N_i is smaller than $\lceil \frac{1+\varepsilon}{2} x \rceil$ is upper bounded by

$$\begin{aligned}
\mathbb{P}\left[N_i < \left\lceil \frac{1+\varepsilon}{2} x \right\rceil\right] &= \sum_{j=1}^{\lceil \frac{1+\varepsilon}{2}x \rceil - 1} \mathbb{P}[N_i = j] \\
&\leq \sum_{j=1}^{\lceil \frac{1+\varepsilon}{2}x \rceil - 1} \binom{|W|}{j} \frac{j^x}{|W|^x} < \sum_{j=1}^{\lceil \frac{1+\varepsilon}{2}x \rceil - 1} \left(\frac{|W|e}{j}\right)^j \frac{j^x}{|W|^x}.
\end{aligned}$$

We observe that when we consider $(\frac{|W|e}{j})^j$ as a function of $j \in \mathbb{R}$ it is strictly monotonically increasing in the range $j \in \mathbb{R} \cap [|W|]$, which implies that the above probability is upper bounded by

$$\begin{aligned}
&\left(\left\lceil \frac{1+\varepsilon}{2} x \right\rceil - 1\right) \left(\frac{|W|e}{\lceil \frac{1+\varepsilon}{2}x \rceil - 1} \right)^{\lceil \frac{1+\varepsilon}{2}x \rceil - 1} \frac{\left(\lceil \frac{1+\varepsilon}{2}x \rceil - 1\right)^x}{|W|^x} \\
&< \left(\frac{1+\varepsilon}{2}x\right) \left(\frac{|W|e}{\frac{1+\varepsilon}{2}x}\right)^{\frac{1+\varepsilon}{2}x} \frac{\left(\frac{1+\varepsilon}{2}x\right)^x}{|W|^x} \\
&= \frac{e^{\frac{1+\varepsilon}{2}x} \left(\frac{1+\varepsilon}{2}x\right)^{\frac{1-\varepsilon}{2}x+1}}{|W|^{\frac{1-\varepsilon}{2}x}} \overset{(3)}{<} \frac{1}{|W|^{\frac{x}{\alpha}}} \overset{(1)}{\leq} \frac{1}{|V|^{\frac{x}{\Delta}}} < 1.
\end{aligned}$$

If there are at least $\lceil \frac{1+\varepsilon}{2} x \rceil$ neighbors in W, then there are at least $\lceil \varepsilon x \rceil$ unique i-neighbors. Therefore, the probability that fewer than $\lceil \varepsilon x \rceil$ nodes in X have a unique i-neighbor for a certain i is strictly smaller than 1.

For any subset of size x, let the random variable F_x denote the event that there are fewer than $\lceil \varepsilon x \rceil$ unique i-neighbors for all $i \in [\Delta]$. Since the random variables $N_1, N_2, \ldots, N_\Delta$ are independent, we immediately get that $\mathbb{P}[F_x] < |V|^{-x}$. Let the random variable F be the event that F_x occurs for any subset of size x for any size $x \in \{2, \ldots, K\}$. The probability of this event is upper bounded by

$$\mathbb{P}[F] \leq \sum_{x=2}^{K} \binom{|V|}{x} \mathbb{P}[F_x] < \sum_{x=2}^{\infty} \binom{|V|}{x} |V|^{-x} = e - 2 < 1.$$

Hence, there is a positive probability that such a randomly chosen graph is a (K, Δ, ε)-matching graph, which proves that such a graph must exist. □

In the remainder of this paper, we use matching graphs with $\varepsilon = 1/8$, which are guaranteed to exist if

$$|W| \geq |V|^{8/\Delta} \text{ and } |W| \geq 3K^3. \qquad (4)$$

Note that the constants in the exponents can be reduced with a more elaborate analysis.

2.2 Reporter-Free Set

A *reporter-free set* is, as the name implies, a set of nodes that are not reporters. Depending on k, different techniques can be employed to compute such a set. A procedure that finds such a set of cardinality x in $\mathcal{O}(k)$ time using one channel has been described in the literature [14]. We now discuss an extension of this procedure, which we call *FindRFS*(x).

Case 1: If k is in the same order of magnitude as x or larger, there exists a constant $c > 0$ such that $x \leq c \cdot k$. Finding a reporter-free set of size x can be accomplished by letting the nodes with identifiers $1, 2, \ldots, k + x$ transmit (on the single channel) if they are reporters or not. This procedure stops after x non-reporters have been found, or the k reporters have been detected and the information exchange problem is solved (and the reporter-free set is thus not needed anymore).

Case 2: Alternatively, we can compute a reporter-free set of size $x \leq n/(k+1)$, which is more suitable for smaller values of k, as follows. The first two nodes (with identifier 1 and 2) are assigned the roles of *leader* and *guard*. All nodes are partitioned into $k+1$ groups based on their identifiers, each containing roughly $n/(k+1)$ nodes. In the first time step, the reporters in the first group (if any) transmit a message on a single predefined channel while the leader is listening unless it is a reporter itself. The guard node also transmits a message containing its identifier if it is not a reporter, otherwise it remains silent. As a consequence, if the leader receives the guard's message, none of the other nodes in the first group are reporters. The leader can then (depending on the guard's message and its own status) announce in the next time step whether the first group is reporter-free and the algorithm terminates. Otherwise, the leader instructs the nodes to continue, in which case all reporters in the second group and also the guard node send a message in the next time step. Again, if the leader only receives the guard's message, it has found a reporter-free set of size $n/(k+1)$, otherwise the same steps are repeated with the third group, fourth group, etc. until a reporter-free group is found. The first x nodes of this group are then assigned to the reporter-free set.

LEMMA 1 (EXTENSION OF LEMMA 5.1 OF [14]). *Procedure FindRFS(x) ensures deterministically that after its completion all nodes know the identifiers of a reporter-free set of size x after $\mathcal{O}(k)$ time steps using one channel if (i) $x \leq c \cdot k$ for some constant c or (ii) $x \leq n/(k+1)$.*

PROOF. Case 1: The correctness of the algorithm is immediate. The number of time steps required is at most $\min\{(c+1)k, n\} \in \mathcal{O}(k)$.

Case 2: For all groups it holds that receiving the guard's message implies that the group does not contain any reporters: For the first group, the guard only transmits if it is not a reporter, and due to possible collisions the leader only receives this message if all other nodes in the first group remain silent, i.e., no node is a reporter. For any group $2, \ldots, k+1$, the guard always sends, and the leader receives this message if there are no reporters in the corresponding group. Since there are $k+1$ groups, there is at least one group that does not contain reporters. Hence, a reporter-free set of size $x \leq n/(k+1)$ is always found. It takes 2 time steps to determine whether a group is reporter-free and to broadcast this information. As there are $k+1$ groups, the number of time steps is upper bounded by $2(k+1) \in \mathcal{O}(k)$. □

COROLLARY 1. *Lemma 1 implies that a reporter-free set of size $\Omega(k+n/k) = \Omega\left(\sqrt{n}\right)$ can be found in $\mathcal{O}(k)$ time slots.*

As we will see in Section 3, our algorithms start by computing such a set. After this computation, in order to simplify the notation, we assume that each node that is not part of the reporter-free set chooses a (potentially) new unique name in the range $1, \ldots, n'$, where n' is the number of nodes not in the reporter-free set minus the number of reporters that have already been detected while computing the reporter-free set. This renaming does not require any communication because the set of nodes in the reporter-free set is globally known.

2.3 Renaming

Renaming is an important concept that is used repeatedly in our algorithms. Initially, the identifiers of the k reporters are in the range $[n] = \{1, \ldots, n\}$. The goal of renaming is to assign new names to the reporters in order to reduce the size of the possible range with reporters so that the reporters can be determined quickly by examining this smaller range. In this section, we describe Procedure *BasicRename*(k,n), which uses matching graphs for efficient renaming.

Let W denote the target namespace, i.e., once *BasicRename*(k,n) terminates, each reporter has a new unique name in W, and let the names in this set be $1, \ldots, |W|$. A prerequisite for *BasicRename*(k,n) is a reporter-free set of size $|W|$, which we assume to be given. As we will see, the namespace W can be chosen to be small enough for all our purposes such that a reporter-free set of size $|W|$ can be computed in $\mathcal{O}(k)$ time using procedure *FindRFS*(x) (Corollary 1). The nodes in the reporter-free set are called *guard nodes*. Moreover, we assume that there are $|W|$ channels $1, \ldots, |W|$. Procedure *BasicRename*(k,n) itself consists of two phases: a *competition phase* and a *conflict resolution phase*.

The goal of the competition phase is to assign a new name in W to each reporter. It is possible that several reporters obtain the same name in this phase. A reporter v competes for a name $i \in W$ by sending its own identifier on channel i. The i^{th} guard node g_i in the reporter-free set listens on channel i. If it receives an identifier, we say that the reporter v that sent this message *won* the competition for name $i \in W$. In this case, g_i adds v's identifier to the list of identifiers that acquired this name. In the subsequent time step, g_i transmits v's identifier on channel i and v listens. This way v is informed that it has won the competition. Each reporter remains *active* until it wins its first competition, i.e., it continues to compete for a name by sending its identifier on the corresponding channel and listen for the retransmission in the subsequent time slot until it succeeds. Once a reporter wins, it becomes *inactive*, which means that it remains silent until the end of the competition phase. The sequence of names that each individual reporter v competes for is determined using a (shared) (K, Δ, ε)-matching graph $G = (V \cup W, E)$, where $K \geq k$. The set V represents the original $n = |V|$ node identifiers, and W represents the target namespace (the new temporary node "names"). Reporter v first competes for its 1-neighbor $\Gamma(v,1)$, and then for its 2-neighbor $\Gamma(v,2)$, etc. If it loses the competition for $\Gamma(v,\Delta)$, it starts again with $\Gamma(v,1)$, i.e., a reporter cycles through its neighbors until it wins. After $\Delta \cdot \lceil \log k / \log(1/(1-\varepsilon)) \rceil$ competitions, the competition phase is over. We show in the proof of Theorem 2 that this number of competitions suffices to guarantee that each reporter indeed wins one of the competitions.

The goal of the conflict resolution phase is to ensure that each reporter obtains a unique name in W. For this purpose, the reporter-free set is partitioned into $\lceil |W|/k \rceil$ groups, each consisting of at most k guard nodes based on their identifiers. Since the identifiers of the nodes in the reporter-free set are known, no communication is necessary for this partitioning. Consider the j^{th} such guard group. The guard node with the smallest identifier starts transmitting the identifiers of the reporters that won a competition for its name on channel j. Once it has transmitted all winning identifiers, the guard node with the next larger identifier starts transmitting on channel j and so on until all guards have transmitted their winners. Each reporter v that won the competition for a name i listens to the communication in the group to which the guard of i belongs and records the identifiers of all reporters that won a competition in this group (includ-

Algorithm 1 Procedure *BasicRename*(k, n): Guard node g_i assigned to channel i

```
// ** competition phase starts
$winnerList := \emptyset$
for $t := 1, \ldots, 9\Delta\lceil\log k\rceil$ do
    receive $message$ on channel $i$
    if identifier $id$ received then
        send $id$ on channel $i$
        $winnerList := winnerList \cup \{id\}$
    else
        send $\perp$ on channel $i$
    end if
end for
$nextGuard := 1$; $winnerId := \perp$

// ** conflict resolution phase starts
for $t := 1, \ldots, 2k-1$ do
    if $nextGuard = i \mod k$ then
        if $winnerList = \emptyset$ then
            $nextGuard := nextGuard + 1$
        else
            $winnerId := \min\{id \mid id \in winnerList\}$
            $winnerList := winnerList \setminus \{winnerId\}$
        end if
        send $[nextGuard, winnerId]$ on channel $\lceil i/k\rceil$
    else
        receive message $[guard, id]$ on channel $\lceil i/k\rceil$
        $nextGuard := guard$
    end if
end for
```

Algorithm 2 Procedure *BasicRename*(k, n): Reporter v

```
// ** competition phase starts
for $t := 1, \ldots, 9\Delta\lceil\log k\rceil$ do
    send $id_v$ on channel $\Gamma(v, (t \mod \Delta) + 1)$
    receive $message$ on channel $\Gamma(v, (t \mod \Delta) + 1)$
    if $id_v$ received then
        $winningChannel := \Gamma(v, (t \mod \Delta) + 1)$
        sleep $2(9\Delta\lceil\log k\rceil - t)$ time slots
        break for-loop
    end if
end for
$winnerList := \emptyset$

// ** conflict resolution phase starts
for $i := 1, \ldots, 2k-1$ do
    receive message $[guard, id]$ on channel $\lceil winningChannel/k\rceil$
    $winnerList := winnerList \cup \{id\}$
end for
$pos := v$'s position in $winnerList$
rename to $(\lceil winningChannel/k\rceil - 1) \cdot k + pos$
```

ing its own). If v listens to the communication in group j and v's identifier is at position $pos \in [k]$ in the ordered list of all received identifiers, node v renames itself using name $(j-1) \cdot k + pos$. After the execution of this algorithm all reporters know their own new name. Note that they know nothing about the new names of the other reporters; what knowledge the other nodes have gained is not considered.

The actions of the guard nodes and the reporters are summarized in Algorithm 1 and Algorithm 2, respectively. Throughout the entire paper, we use the convention that time only passes in the pseudo code if the node *waits* until a certain time or if it sends or receives a message. More precisely, if a node sends/receives a message at time t, the time is $t+1$ *after* this operation, i.e., receiving is non-blocking (in the sense that the operation lasts exactly one time step). If a node *waits* until time t, it is exactly time t after this operation. The following theorem summarizes the properties of procedure *BasicRename*(k, n).

THEOREM 2. *Given a $(K, \Delta, 1/8)$-matching graph $G(V \cup W, E)$, where $|V| = n$ and $K \geq k$, and a reporter-free set of size $|W|$, BasicRename(k, n) assigns a unique name in $[|W|]$ to all k reporters in time $\mathcal{O}(\Delta \cdot \log(k) + k)$ using $|W|$ channels.*

PROOF. The time complexity follows from the description of the algorithm: The competition phase takes $O(\Delta \cdot \log k)$ time steps and the conflict resolution phase takes $2k-1$ time steps, i.e., the overall time complexity is $\mathcal{O}(\Delta \cdot \log(k) + k)$ as claimed. While the competition phase requires $|W|$ channels, $\lceil W/k \rceil$ channels suffice for the conflict resolution phase. Thus, $|W|$ channels are used in total.

It remains to prove the correctness of the algorithm. For each competition it holds that if there is exactly one contending node v for channel i, then v wins the competition for name i; otherwise, all competitors lose. This property holds due to the synchronous nature of our model and the fact that the guard node only receives a message if exactly one node competes for the channel at a given point in time.

Since the nodes use a $(K, \Delta, 1/8)$-matching graph, it holds that for some $i \leq \Delta$ at least $1/8$ of the $k \leq K$ reporters have a unique i-neighbor and thus have competed for a channel successfully after Δ competitions. Hence, at most $\lfloor 7k/8 \rfloor$ reporters are active after Δ competitions, $1/8$ of which win in the next Δ competitions and so on, i.e, after $i\Delta$ competitions at most $\lfloor (7/8)^i \cdot k \rfloor$ reporters are left. Consequently, there are no active reporters left after $\Delta \cdot \lceil (\log k / \log(8/7)) \rceil < 6\Delta \cdot \log k$ competitions as required. At the end of the competition phase, all reporters listening to the same group j choose distinct names in the range $[(j-1)k, jk]$. Assume that v listening to group j and v' listening to group $j' \neq j$ choose the same name. Without loss of generality, assume that $j > j'$. In this case, reporter v chooses the name $(j-1)k + p \geq j'k + p = (j'-1)k + (k+p)$ for some p, implying that the position in the ordered list of all received identifiers for v' is at least $k + p \notin [k]$, a contradiction. □

Note that the nodes are not required to store a matching graph for all possible values of k and Δ. As we will see in the subsequent sections, we only use matching graphs if k is polylogarithmic in n. Since we do not optimize constants, it hence suffices to store $\mathcal{O}(\log\log n)$ matching-graphs for $k = 2, 4, 8, 16$ and so on. The right choice of Δ for these graphs is discussed in Section 3.

2.4 Information Propagation on Trees

This building block is a procedure that disseminates information over a binary tree. This technique is used in our algorithms for two different purposes: In one algorithm, it is used to inform one node, the root of the tree, about all k reporters, and in the other algorithm, the tree is used to disseminate information about the number of collisions recorded at the leaves. We consider trees containing $N \leq n$ nodes of the network whose identifiers are known to all other nodes participating in the tree-algorithm such that each node can determine its positions in the tree accordingly.

Lemma 2. *The time complexity of disseminating b items on a balanced binary tree of N nodes is $\mathcal{O}(b + \log N)$ using $\mathcal{O}(N)$ channels.*

Proof. In the standard synchronous message-passing model, a straightforward algorithm to gather information at the root is to forward each information item to the parent, and potentially receive information items from the children, in every time step. The time complexity of this algorithm when forwarding b items to the root in a tree of height h is $b + h - 1$. Unfortunately, this simple scheme does not work in our setting because a) a node cannot send and receive simultaneously and b) it cannot listen to different nodes at the same time. However, these problems can be solved by splitting each round into four sub-rounds and ordering the actions in these sub-rounds according to the level in the tree: Every node v_i in the tree has its own channel i. If a node is at an even (or odd) level and it is the left child of its parent, it sends on channel i in the first sub-round. If it is the right child, it sends in the second sub-round. In the third and fourth sub-round, it listens to transmissions from its left and right child on their channels, respectively. If a node is at an odd (or even) level, it first listens to its children on their channels and transmits either in the third or fourth sub-round on its own channel depending on whether it is its parent's left or right child. It is not hard to see that the four sub-rounds simulate a single round in the message passing model and that there are no conflicts.

Note that the same bound on the time complexity holds if information is disseminated from the root to the leaves of the tree. If the nodes are required to forward an aggregate of the information in their subtree, such as, e.g., the sum of nodes in its subtrees, every node only needs to send a message to its parent once (upon receiving the necessary information from its children). The structure of the tree is based on the participating node identifiers, i.e., it is known a-priori; hence, all nodes in the tree can compute their schedule locally without any communication. As the height of a binary tree with N nodes is $\mathcal{O}(\log N)$ and every node uses its own channel for communication, the statement of the lemma follows. □

3. ALGORITHMS

Having discussed the basic building blocks, we now describe our information exchange algorithms. The algorithms presented in this paper are composed of two phases, a *scheduling phase* and a *broadcast phase*. In the scheduling phase, the reporter nodes (and some of the other nodes) exchange messages on the available channels in order to derive a schedule for the broadcast phase. This schedule defines an injective function from the set of reporters to the $\mathcal{O}(k)$ time slots of the broadcast phase. In other words, assuming that the first time slot of the broadcast phase occurs at the logical time 1, every reporter is assigned a unique time slot in the range $[C \cdot k]$ for some constant $C > 1$ after the scheduling phase. In the broadcast phase, each reporter transmits its information in its assigned time slot on channel 1 while all other nodes listen. The time complexity of the broadcast phase is thus $\mathcal{O}(k)$, and it guarantees that all nodes know all information items in the end using only one channel. Hence, the time complexity of an algorithm and the number of channels used depend on the scheduling phase only. Moreover, since the broadcast phase always works the same way, it suffices to discuss the scheduling phase of each algorithm. In order to guarantee that the time complexity of the scheduling phase is always $\mathcal{O}(k)$ we apply different techniques for different values of k.

3.1 Algorithm $\mathcal{A}_S$ for $k \leq \frac{1}{6}\log n$

For small values of k, we propose an information exchange algorithm, denoted by $\mathcal{A}_S$, that uses matching graphs and the procedures described in the previous section.

The algorithm works as follows. First, we determine a reporter-free set of size $n/(k+1)$ using *FindRFS* (in time $O(k)$, see Corollary 1). Subsequently, *BasicRename*(k, n) is executed using a $(k, \Delta, 1/8)$-matching graph for which $\Delta := \lfloor 10k/\log k \rfloor$ and $|W| := \lceil |V|^{8/\Delta} \rceil$. In the proof of Theorem 3, we show that such a graph exists. In the next step, a case distinction is required.

Case 1: If $|W| \leq 2^{6k}$, we use a binary tree containing $|W|$ nodes from the reporter-free set that have been computed with *FindRFS*. The tree is built up layer by layer based on the nodes' identifiers: The node with the smallest identifier is the root, the node with the second smallest identifier is its left child, the node with the third smallest is its right child, etc. Note that no communication is necessary to build this tree as the nodes in the reporter-free set and their identifiers are known to all nodes in the network. The identifiers of the k reporters are forwarded to the root as described in Section 2.4. The root can then distribute all identifiers of reporters in the next k communication rounds.

Case 2: If $|W| > 2^{6k}$, *BasicRename*(k, n) is executed again using the new names as the "input-identifiers". We refer to the new set of names as V' (which is identical to W). The same number Δ of neighbors is used, and $|W'| := \lceil |V'|^{8/\Delta} \rceil$. Again, the existence of such a graph is shown in the proof. Afterwards, all distinct k-element subsets of W' are assigned to distinct nodes in the reporter-free set. This assignment can be computed locally as all nodes in the reporter-free set are known. Each reporter-free node that is assigned at least one subset is called a *listener* (we will get back to this assignment in more detail later). A listener may be assigned at most two distinct subsets.

In the next step, each reporter transmits its identifier on the channel that corresponds to its unique name in W', during the next $2k$ time slots. Simultaneously, each listener listens for one time slot on each of the channels in its first assigned subset during the first k time slots, and then, if it is assigned a second subset, it listens on each channel in the second assigned subset for one time slot.

Finally, the listener receiving identifiers in each of the k communication rounds for a given assigned subset informs the remaining nodes on channel 1 about the reporters in the next k communication rounds. In the proof of Theorem 3 we show that there is one unique listener broadcasting this. Therefore no collisions occur and all nodes know the identifiers of all reporters. Thus the information items can easily be disseminated in the broadcast phase. We get the following result.

Theorem 3. *Algorithm $\mathcal{A}_S$ solves the information exchange problem for $k \leq \frac{1}{6}\log n$ in time $\mathcal{O}(k)$ using $\mathcal{O}(n^{\log(k)/k})$ channels.*

Proof. We start by proving the existence of the matching graph used in the first step. Since

$\Delta \leq 10k/\log k$ and $k \leq \frac{1}{6}\log n$, we have that $|V|^{8/\Delta} \geq n^{\frac{4}{5}\frac{\log k}{k}} \geq n^{24/5\frac{\log k}{\log n}} = k^{24/5} \overset{k\geq 2}{>} 3k^3$. This bound together with Theorem 1 implies that such a graph indeed exists. According to Theorem 2, the time required for this renaming is bounded by $\mathcal{O}(\Delta\log k + k) = \mathcal{O}(k)$.

Case 1: Corollary 2 states that $|W| - 1 \in \mathcal{O}(n^{\log(k)/k})$ channels are required, i.e., the same number as for the renaming. The bound $|W| \leq 2^{6k}$ entails that the time complexity is bounded by $4(k+h-1) \leq 4(k+\log|W|-1) < 28k \in \mathcal{O}(k)$. Since the renaming also takes $\mathcal{O}(k)$ time, the total time complexity is $\mathcal{O}(k)$ as claimed. The correctness follows from the correctness of the renaming and the propagation protocol on trees.

Case 2: Since $|W| > 2^{6k}$, we get that $|V'|^{8/\Delta} = |W|^{8/\Delta} > 2^{48k/\Delta} \geq 2^{24\log(k)/5} = k^{24/5} \overset{k\geq 2}{>} 3k^3$. Thus, Theorem 1 again implies that there is a matching graph where $|W'| = \lceil |V'|^{8/\Delta} \rceil$.

We now show an upper bound on $|W'|$. As $|W'| > 3k^3 \geq 24$, we have that $|W'| < |V'|^{8/\Delta} + 1 < |V'|^{8/\Delta} + \frac{1}{24}|W'|$ and thus

$$\begin{aligned} &\text{a)} \quad |W'| < \frac{24}{23}|V'|^{8/\Delta}, \qquad \text{b)} \quad |V'| < \frac{24}{23}n^{8/\Delta} \\ &\text{c)} \quad \Delta > \frac{10k}{\log k} - 1 > \frac{28}{3}\frac{k}{\log k} \end{aligned} \tag{5}$$

since $|V'| > 3k^3 \geq 24$, with a similar argument as above, and the fact that $k/\log k$ is larger than $3/2$ for all k implies the third inequality. If we combine these bounds, we get that

$$\begin{aligned} |W'| &\overset{(5a,b)}{<} \left(\frac{24}{23}\right)^{1+8/\Delta} n^{(8/\Delta)^2} \\ &\overset{(5c)}{<} \left(\frac{24}{23}\right)^{1+\frac{6\log k}{7k}} n^{\left(\frac{6\log k}{7k}\right)^2} < \left(\frac{24}{23}\right)^{\frac{11}{7}} n^{\frac{1}{k}}. \end{aligned} \tag{6}$$

Hence, the number of k-element subsets of W' is upper bounded by $\binom{|W'|}{k} < \frac{|W'|^k}{k!} \overset{(6)}{<} \frac{\left(\frac{24}{23}\right)^{\frac{11}{7}k} n}{k!} \overset{k\geq 2}{<} 2\frac{n}{k+1}$. Thus, the subsets can be assigned in such a way that each node in the reporter-free set is assigned at most two subsets. Therefore, a unique listener determines the identifiers of the k reporters within $2k$ time slots, which proves the correctness of the algorithm in this case. Since both the second renaming and the computation of the correct k-element subset need fewer channels than the first renaming, the number of channels required is $|W| = |V'| \in \mathcal{O}(n^{\log(k)/k})$. The second renaming also takes $\mathcal{O}(\Delta\log k + k) = \mathcal{O}(k)$ time, and the computation of the subset requires $2k$ time slots. Hence, Algorithm $\mathcal{A}_S$ needs $\mathcal{O}(k)$ time slots in total for Case 2 as well. □

3.2 Algorithm $\mathcal{A}_L$ for $k \in \left(\frac{1}{6}\log n, n - 2\lceil\log(n)\rceil\right)$

For $k \in \left(\frac{1}{6}\log n, n - 2\lceil\log(n)\rceil\right)$ we use a different approach than in Algorithm $\mathcal{A}_S$. We start with a high-level descriptions of Algorithm $\mathcal{A}_L$ and its main building block called an *epoch* which is discussed in more detail in Section 3.2.1 and Section 3.2.2.

High-level description of Algorithm $\mathcal{A}_L$: After some preprocessing, Algorithm $\mathcal{A}_L$ executes a sequence of *epochs*. The goal of an epoch is to identify some previously unknown reporters. The number of epochs that need to be carried out depends on the number k of reporters. If k is smaller than $\log(n)\cdot\log\log(n)$ (Case 1), we run a procedure called *DetectFraction* several times, to reduce the number of unknown reporters. During the execution of *DetectFraction*, temporary names are assigned to the reporters using matching graphs. The range of these temporary names is smaller than the range of the identifiers of the nodes. Subsequently, multiple epochs are executed using these temporary names. Due to the smaller range of temporary names, not all temporary names are unique and thus an epoch might detect only a few reporters because the messages of reporters with the same temporary names collide. Procedure *DetectFraction* executes Δ epochs, one for each neighbor index of the used matching graph, which will be specified later. This allows us to guarantee that in at least one of these epochs the number of unknown reporters is reduced significantly. Otherwise (Case 2), one epoch is sufficient to detect all reporters. After $\mathcal{O}(k)$ time slots all reporters are known in both cases.

High-level description of an epoch: Given k' reporters with temporary names in a temporary namespace $[n']$ that were not detected in a previous epoch, an epoch finds and broadcasts the identifiers (in $[n]$) of reporters with a unique temporary name in $[n']$. This is achieved as follows. First, the temporary namespace is partitioned into groups. The assignment of temporary names to the groups is then refined in a number of phases. At the beginning of a phase, each group comprises a range of temporary names and each reporter's temporary name belongs to exactly one group. After a phase, the number of groups is the same but the temporary names belonging to a group change. Moreover, each group contains fewer temporary names than before and it still holds that each reporter's temporary name belongs to exactly one group.

Since the number of temporary names per group decreases in each phase, the number of temporary names in each group is one after a certain number of phases. If this left-over temporary name belongs to exactly one reporter, we can determine whether this node is a reporter and broadcast its identifier to the whole network. Multiple channels are used to ensure that the time complexity of each epoch is small enough.

3.2.1 Description of Algorithm $\mathcal{A}_L$

Algorithm $\mathcal{A}_L$ uses $2c := 2\tau\cdot\lceil\log(n/k)\rceil$ channels, for some τ defined later, and a reporter-free set of $2c$ nodes. Among these $2c$ nodes, c nodes are *master nodes* denoted by $\{m_1,\ldots,m_c\}$, the other c nodes are *helper nodes* $\{h_1,\ldots,h_c\}$. Master m_j communicates on channel j unless stated otherwise. As mentioned in the high-level description, $\mathcal{A}_L$ executes one or more epochs to detect unknown reporters with unique temporary names in $[n']$. We distinguish between two cases.

Case 1: If $k < \log(n)\cdot\log\log(n)$, we set $\tau := \lceil\log^\rho n\rceil$ for some constant $\rho > 0$. Next, we run Procedure *DetectFraction* (Line 9) multiple times. This procedure comprises several epochs and detects a constant fraction of the k' remaining unknown reporters. We repeatedly execute *DetectFraction* until at most $\frac{1}{6}\log n$ reporters are unknown (Lines 8–10). Once $k' \leq \frac{1}{6}\log n$, Algorithm $\mathcal{A}_S$ is called to detect the remaining unknown reporters (Line 11).

Procedure *DetectFraction* uses a (k,Δ,ε)-matching graph G with $\Delta = \lceil\log\log n\rceil$, $\varepsilon = 1/8$, $|V| = n$, $|W| = n' := \lceil n^{8/\Delta}\rceil$ (Line 13). Theorem 1 proves the existence of such a graph. This graph is utilized to compute the temporary names that are used throughout one of the Δ epochs

Algorithm 3 Sketch of Algorithm $\mathcal{A}_L$

```
 1: if $k < \log(n) \cdot \log\log n$ then
        // ** Case 1:
 2:     $\tau := \lceil \log^{\rho} n \rceil$; $c := \tau \cdot \lceil \log(n/k) \rceil$; $k' := k$
 3:     compute $c$ master and $c$ helper nodes
 4:     while $k' > \frac{1}{6} \log n$ do
 5:         run $DetectFraction()$
 6:     end while
 7:     run Algorithm $\mathcal{A}_S$ for remaining reporters
 8: else
        // ** Case 2:
 9:     $\tau := 2$; $c := \tau \cdot \lceil \log(n/k) \rceil$
10:     compute $c$ master and $c$ helper nodes
11:     run epoch on temporary namespace $[n'] := [n]$
12: end if

    Procedure $DetectFraction$:
13: $G := (k, \lceil \log\log n \rceil, 1/8)$-matching graph
                with $|W| = n' = \lceil n^{8/\lceil \log\log n \rceil} \rceil$
14: for $i := 1, \ldots, \lceil \log\log n \rceil$ do
15:     temporary names according to $i^{th}$ neighbor in $G$
16:     run epoch with these temporary names in $[n']$
17:     $k' := k' - $#reporters detected in epoch
18: end for
```

of *DetectFraction*; in the i^{th} epoch, the unknown reporters use temporary names according to their i^{th} neighbor in the (k, Δ, ε)-matching graph. In each of the epochs, a few reporters are able to tell their original identifiers to the rest of the nodes. The temporary name l in $[n']$ of a node v (called v_l) may not always be unique. Although there can be several nodes with the same temporary name, we write v_l when we refer to one of these nodes.

Case 2: If $k \geq \log(n) \cdot \log\log(n)$ (Lines 2–4 in Algorithm 3), only one epoch with temporary namespace $[n'] := [n]$ and $\tau := 2$ is used (Line 2). Before executing this epoch (Line 4), Algorithm $\mathcal{A}_L$ starts the preprocessing by computing a reporter-free set of size $2c$, and assigns master and helper nodes accordingly (Line 3).

In Section 3.2.3 we prove that Algorithm $\mathcal{A}_L$ successfully identifies the k reporters in $\mathcal{O}(k)$ time.

3.2.2 *Description of an Epoch*

Throughout the entire execution of an epoch, each unknown reporter v_i is a member of exactly one out of at most $\tau \cdot k'$ disjoint groups. Observe that we defined groups to contain temporary names. For simplicity we also treat them as if they contained nodes: a node is a member of a group if its temporary name is contained in this group.

As mentioned earlier, an epoch consists of several phases. Group membership of a name/node remains the same throughout the course of a phase but it may change at the end of each phase. Therefore we introduce a *group number* $g(i)$ for each temporary name i, which indicates to which group node v_i belongs due to its temporary name i in the current time slot. I.e., $g(i) = j$ indicates that v_i is in the j^{th} group, where $j \in \{1, \ldots, \tau \cdot k'\}$. In phase 1, a node v_i belongs to group $g(i) = \lceil i \cdot \tau \cdot k'/n' \rceil$, i.e., any reporter with a temporary name less than $\lceil n'/(\tau \cdot k') \rceil$ belongs to group number 1, reporters among the next $\lceil n'/(\tau \cdot k') \rceil$ nodes to group number 2 and so on. In other words, at the beginning of an epoch the temporary namespace $[n']$ is partitioned into disjoint ranges of length at most $\lceil n'/(\tau \cdot k') \rceil$, each associated with one group.

In every phase, reporter v_i also stores the range $r(i) := [l, r]$ of temporary names lying in its group's range, i.e., l is the lowest temporary name such that $g(l) = g(i)$ and r is the largest temporary name such that $g(r) = g(i)$. This range is updated each time v_i's group changes.

The algorithm ensures that at the end of an epoch, if v_i has a unique temporary name i, it is the only reporter in the group with identifier $g(i)$. Otherwise, group $g(i)$ contains either no reporters or more than one reporter.

Now we specify how reporters and masters communicate. Each master communicates with reporters belonging to at most $s := \lceil \tau \cdot k'/c \rceil = \lceil k'/\lceil \log(n/k) \rceil \rceil$ distinct groups: m_1 communicates on channel 1 with reporters in the first s groups, m_2 communicates on channel 2 with reporters in the next s groups, and so on, i.e., master m_j communicates on channel j with reporters in the group ℓ if $j = \lceil \ell/s \rceil$. We say that these groups *belong* to master m_j. In each phase, reporter v_i only communicates with the master $m_{\lceil g(i)/s \rceil}$ to which the group number $g(i)$ belongs, i.e., it uses channel $\lceil g(i)/s \rceil$ for communication. Furthermore, each reporter v_i stores its group's position $p(i) := g(i) \mod s$ in the sorted list of all groups that belong to its group's master. The position is used to determine the communication time slot between the nodes in a specific group and its master. Note that a master m_j always communicates with the same groups, but these might contain different reporters in each phase.

Since every phase is identical, it suffices to study the operation of a single phase to understand an epoch. The goal of a phase is to reduce the number of nodes contained in each group. Recall that at the beginning of a phase, each group contains a certain number of temporary names. Each reporter is contained in some group due to its temporary name. After a phase, the number of groups is the same, the temporary names contained in a group might differ, but each group contains a smaller temporary namespace than before. It still holds that each reporter is contained in some group via its temporary name.

A phase consists of three parts. Since the helper nodes only operate in the first part of any given phase and act at the same times on the same channels as their masters, pseudo code for the helpers is omitted. The execution of a phase is described for masters in Algorithm 4 and for reporters in Algorithm 5. As every phase takes the same number of time steps, we can assume for ease of notation that every phase starts at a (logical) time $t = 0$. Note that master, helper and reporter nodes execute their specific code at the same time in synchronized time slots.

We now proceed to describe the steps of a phase in greater detail. The three parts of a phase work as follows.

Part 1 of a phase: Detect all groups that contain at least one reporter. See Lines 1–7 in Algorithm 4 and Lines 1–4 in Algorithm 5. We denote the set of groups in which m_j detects a reporter by $\mathcal{C}_j$. To compute $\mathcal{C}_j$, each master m_j checks each of its $s = \lceil \tau \cdot k'/c \rceil$ groups for reporters one after another. Any master m_j listens and its helper h_j sends an arbitrary message on channel j for s time steps, while each reporter v_i sends a message on channel $\lceil g(i)/s \rceil$ to its master at time $p(i)$. Remember that v_i is in the $p(i)^{th}$ group that belongs to $m_{\lceil g(i)/s \rceil}$ at that time. Thus, we can conclude that if a master m_j receives a message at a time $t + q$, $q \in \{1, \ldots, c\}$, on channel j, it must be the message sent by helper h_j, i.e., m_j learns that there are no reporters

Algorithm 4 Code executed by master m_j during a phase within Algorithm $\mathcal{A}_L$.

```
    // ** Part 1:
 1: $\mathcal{C}_j := \emptyset$
 2: for $t := 0 \dots s-1$ do
 3:     receive message on channel $j$
 4:     if no message received then
 5:         $\mathcal{C}_j := \mathcal{C}_j \cup \{t\}$;   // ** collision in the $t^{th}$ group
 6:     end if
 7: end for
    // ** Part 2:
 8: $coll_{j-1} :=$ sumOfCollisionsAtMasters($m_1, \dots, m_{j-1}, |\mathcal{C}_j|$)
 9: wait until $t = s - 1 + 8\lceil \log c \rceil$
    // ** Part 3:
10: for $\gamma := 0 \dots s-1$ do
11:     if $\gamma \in \mathcal{C}_j$ then
12:         send $coll_{j-1}$ on channel $\gamma$
13:         $coll_{j-1} := coll_{j-1} + 1$
14:     end if
15: end for
```

in its q^{th} group. On the other hand, if no message is received, a collision must have occurred due to some node v_i with $\lceil g(i)/s \rceil = j$ and $p(i) = q$, implying that the q^{th} group contains at least one reporter. Thus, each masters m_j knows its set $\mathcal{C}_j$ after Part 1.

Part 2 of a phase: Redefine groups. Each master m_j computes for each group it is responsible the range of temporary names that m_j will assign in Part 3 to the (new) groups that result from splitting any group in which m_j detected a collision in Part 1. See Line 8 in Algorithm 4. First, each master m_j (with $j > 1$) has to learn the number $coll_{j-1} := \sum_{p=1}^{j-1} |\mathcal{C}_p|$ of all collisions that occurred at the masters $m_1, \dots, m_{j-1}$ (thus $coll_{j-1}$ is at most $(j-1) \cdot s$). This is achieved by calling the function *sumOfCollisionsAtMasters*, which uses a balanced binary tree (see Section 2.4) of depth $\log c$ to compute $coll_{j-1}$ for master m_j (for all $j \in \{1, c-1\}$ simultaneously.) In this tree, the c masters are the leaves and $c-1$ helpers are used as the inner nodes. By aggregating the number of collisions $|\mathcal{C}_1|, \dots, |\mathcal{C}_{c-1}|$ from the leaves, each inner node h knows the number of collisions that occurred at the masters in its left and right subtree. Let ℓ_h denote the number of collisions in its left subtree. Aggregating the total sum of collisions at the root (e.g., h_1) takes at most $4\lceil \log c \rceil$ time steps and at most c channels as discussed in Section 2.4. The number of collisions at the leaves (masters) in the left subtree of the root is then computed as follows: The root sends 0 to its left child and the number ℓ_{h_1} of collisions in its left tree to its right child. Any inner node h that receives x from its parent sends x to its left child and $x + \ell_h$ to its right child. It is easy to verify that each master m_j receives exactly the number of collisions $\sum_{p=1}^{j-1} |\mathcal{C}_p|$ that were reported by the masters $m_1, \dots, m_{j-1}$ since these are positioned to m_j's left in the tree. This is exactly $coll_{j-1}$. Propagating this information from the root to the leaves takes at most $4\lceil \log c \rceil$ time steps as shown in Section 2.4.

Part 3 of a phase: Each group that contains at least one reporter is split into τ (new) groups. Group identifiers are reassigned to these (new) groups. See Lines 9–15 in Algorithm 4 and Lines 5–12 in Algorithm 5. First, each master m_j creates $\tau \cdot |\mathcal{C}_j|$ new groups: τ new groups for each of the $|\mathcal{C}_j|$ groups in which it detected a collision. Since node m_j knows that the total number of collisions that occurred at the masters $\{m_1, \dots, m_{j-1}\}$ is $coll_{j-1}$, it can assign the group numbers from $\tau \cdot coll_{j-1}$ to $\tau \cdot (coll_{j-1} + |\mathcal{C}_j|) - 1$ to these new groups.

Algorithm 5 Code executed by reporter v_i in group $g(i)$ with range $r(i) = [l, r]$ during a phase within Algorithm $\mathcal{A}_L$.

```
    // ** Part 1:
 1: $c(i) := \lceil g(i)/s \rceil$ // ** channel of this group
 2: $t := 0$ // ** beginning of phase
 3: wait until $t = p(i)$ // ** position of this group
 4: send "message" on channel $c(i)$
    // ** Part 2: reporters sleep
    // ** Part 3:
 5: wait until $t = s - 1 + 8\lceil \log c \rceil + p(i)$
    // ** The master of $v_i$'s group is $m_j$ with $j := \lceil g(i)/s \rceil$
 6: receive $coll_{j-1}$ on channel $c(i)$
 7: compute $j$, s.t., $v_i \in [l + j(r-l)/\tau, l + (j+1)(r-l)/\tau]$
 8: $g(i) := \tau \cdot coll_{j-1} + j$
 9: $r(i) := [l + j(r-l)/\tau, l + (j+1)(r-l)/\tau]$
10: $p(i) := g(i) \mod s$ // ** update position of this
                              group in the list of its (new) master
```

The reporters are informed about their new group numbers by executing a code sequence similar to Part 1 again, in which the roles of sender and receiver are switched, and the helper nodes remain silent. That is, the masters broadcast group identifiers and the reporters receive them. Afterwards, temporary names are reassigned to the new groups: Each new group with group number in $[\tau \cdot coll_{j-1}, \tau \cdot (coll_{j-1} + |\mathcal{C}_j|) - 1]$ receives a $1/\tau$ fraction of the temporary names of an original group that is split. A reporter v_i changes its group membership $g(i)$ to $\tau \cdot coll_{(\lceil g(i)/s \rceil)-1} + \ell$ if its temporary name i lies in the range of the ℓ^{th} new group that its master $m_{\lceil g(i)/s \rceil}$ created. Each reporter knows what to change by listening to the corresponding channel as indicated in the pseudocode of Algorithm 5. Each reporter is able to determine which master its group belongs to in the next phase by performing the computations described earlier.

3.2.3 Analysis of Algorithm $\mathcal{A}_L$

First, we study the time and channel complexity of an epoch.

LEMMA 3. *Given a reporter-free set of size $2c$, one epoch of Algorithm $\mathcal{A}_L$ for k' unknown reporters detects all reporters with unique temporary names in the temporary namespace $[n']$ using $2c = 2\tau \cdot \lceil \log(n/k) \rceil$ channels in time $\mathcal{O}\left(\left(\frac{k'}{\log(n/k)} + \log c\right) \cdot \left(\frac{\log \frac{n'}{\tau \cdot k'}}{\log \tau}\right)\right)$.*

PROOF. Correctness: Since every master has its own channel to communicate with its groups, there are no collisions between the masters. Consider master m_j and its groups. According to the description, only the reporters of the same group send at the same time together with the helper h_j, i.e., collisions among reporters with distinct temporary names can only occur if two or more reporters are in the same group, which proves that the groups with reporters with distinct temporary names are detected correctly. The information propagation on the tree is correct as discussed in Section 2.4, and thus the masters can successfully narrow down the ranges of the temporary names that belong to at least one reporter in each phase and inform the reporters about their new group memberships.

Time complexity: The execution of the tree algorithm in each phase depends on the height of the tree, which is logarithmic in the number of masters and thus takes time $\mathcal{O}(\log c)$. Hence, the time complexity of a single phase is

$$\mathcal{O}(s + \log c) = \mathcal{O}\left(\frac{k'}{\log(n/k)} + \log c\right),$$

taking the length of the **for** and the **while** loops into account. Since the maximum number of temporary names contained in a group in phase 0 is $\lceil n'/(\tau \cdot k') \rceil$, which is divided by τ in each phase, one epoch consists of at most $\left\lceil \log_\tau \left\lceil \frac{n'}{\tau \cdot k'} \right\rceil \right\rceil \in \mathcal{O}\left(\frac{\log \frac{n'}{\tau \cdot k'}}{\log \tau}\right)$ phases. Hence the time complexity of one epoch is $\mathcal{O}\left(\left(\frac{k'}{\log(n/k)} + \log c\right) \cdot \left(\frac{\log \frac{n'}{\tau \cdot k'}}{\log \tau}\right)\right)$. □

THEOREM 4. *Algorithm $\mathcal{A}_L$ solves the information exchange problem in time $\mathcal{O}(k)$ using*

Case 1: $\mathcal{O}(\log^{1+\rho} n)$ channels for some constant $\rho > 0$ if $k \in (\frac{1}{6}\log n, \log(n) \cdot \log\log n)$.

Case 2: $\mathcal{O}(\log(n/k))$ channels if $k \geq \log(n) \cdot \log\log n$.

PROOF. Note that for both cases a reporter-free set of size $2c$ can be found in time $\mathcal{O}(k)$ as shown in Lemma 1, which is a prerequisite for the correctness and the time complexity.

Case 1: Correctness: Note that messages sent by reporters with the same temporary name in $[n']$ always collide and their groups are detected, and therefore their identifiers cannot be detected at the end of an epoch. However, their participation does not disturb the course of the algorithm. Due to the property of the $(k, \Delta, 1/8)$-matching graphs used, at least 1/8 of the unknown reporters have unique temporary names during some epoch, and will be detected in this epoch. We conclude that the number of unknown reporters is reduced in each call of the Procedure *DetectFraction* and only an $(1 - 1/8) = 7/8$-fraction of the reporters that were unknown before calling *DetectFraction* remain unknown. At some point, all but $\frac{1}{6}\log n$ reporters have been detected. These remaining reporters are then determined by Algorithm $\mathcal{A}_S$, and it follows that Algorithm $\mathcal{A}_L$ correctly identifies all reporters.

Complexity: It holds that at least $\frac{1}{8}k'_{old}$ of the k'_{old} reporters that were unknown before executing *DetectFraction* are detected during Procedure *DetectFraction*. Procedure *DetectFraction* is called again if the new number k'_{new} of still unknown nodes is $k'_{new} > \frac{1}{6}\log n$.

CLAIM 1. *If there are k' reporters, DetectFraction needs time $\mathcal{O}(k')$ and $\mathcal{O}(\log^{1+\rho} n)$ channels.*

PROOF. By applying Lemma 3 with $n' := \lceil n^{8/\lceil \log\log n \rceil} \rceil$ and $\tau := \lceil \log^\rho n \rceil$, we can deduce that $2c = 2\tau \cdot \lceil \log(n/k) \rceil \in \mathcal{O}\left(\log^{1+\rho} n\right)$ channels suffice and the time complexity of one epoch is

$$\begin{aligned}
&\mathcal{O}\left(\left(\frac{k'}{\log(n/k)} + \log c\right) \cdot \left(\frac{\log \frac{n'}{\tau \cdot k'}}{\log \tau}\right)\right) \\
&= \mathcal{O}\left(\frac{k'}{\log(\log n) \cdot \log \tau} + \frac{\log(c) \cdot \log n}{\log(\log n) \cdot \log \tau}\right) \\
&= \mathcal{O}\left(\frac{k'}{(\log\log n)^2} + \frac{\log n}{\log\log n}\right),
\end{aligned}$$

which is $\mathcal{O}\left(\frac{k'}{\log\log n}\right)$ due to the range of k that we consider in this case. In total, $\Delta = \log\log n$ epochs are executed and the claim follows. □

When starting with $k' := k$, in each call of Procedure *DetectFraction* at least 1/8 of the remaining reporters are detected. After i calls, k' is reduced to at most $(7/8)^i k$. Claim 1 proves that k' is less than $\frac{1}{6}\log n$ after at most $\sum_{i=0}^{\log_{7/8}(k - \frac{1}{6}\log n)} \mathcal{O}((7/8)^i k) = \mathcal{O}(k)$ time slots and we can apply Algorithm $\mathcal{A}_S$.

Case 2: The correctness of Algorithm $\mathcal{A}_L$ for the second range of k follows directly from Lemma 3. Analogously, both the channel and the time complexity can be derived by applying Lemma 3 with $n' = n$, $\tau = 2$ and $k' = k$: We get a bound on the channel complexity of $\mathcal{O}(\log(n/k))$, and a time complexity of $\mathcal{O}\left(\left(\frac{k}{\log(n/k)} + \log c\right) \cdot \log(n/k)\right) = \mathcal{O}(k)$. □

4. LOWER BOUND

In this section, we prove a lower bound on the number of channels required to achieve a time complexity of $\mathcal{O}(k)$ in a deterministic setting. Again, we assume that $k > 1$ as the information exchange problem is trivial for $k \leq 1$. Throughout this section, the nodes are allowed to send and listen on all c channels *at the same time*; moreover, the nodes can detect collisions, i.e., they have the ability to distinguish between a collision and a transmission-free channel. Thus, the lower bound holds in a stronger model than the algorithms we described.

We proceed by first showing that for any deterministic algorithm $\mathcal{A}$ there is an assignment of the k reporters such that it takes at least a certain number of communication rounds to detect them given a specific number of channels c. As we will see, this result directly implies a lower bound on the number of channels required to guarantee a time complexity of $\mathcal{O}(k)$. In order to prove that there is such an assignment, we introduce the notion of *potential reporters*, which are all the nodes that may be reporters after a certain number of communication rounds. In particular, this means that an "adversary" may still decide for each node among the potential reporters whether or not it is a reporter subject to the constraint that k nodes must be reporters. Formally, let R^ℓ denote the set of all potential reporters after $\ell \geq 0$ rounds of communication. Naturally, we have that $R^0 = V$. The following lemma, which states that there is an assignment of reporters for which the number of potential reporters is reduced by at most a factor of $(c+1)^2$ per communication round, is key for the arguments used later.

LEMMA 4. *Assume that all nodes* only *know that the k reporters are in R^ℓ, where $|R^\ell| \geq (c+1)^2 \cdot (k+2)$, after $\ell \geq 0$ rounds of communication. The reporters can be assigned in such a way that after $\ell + 1$ rounds of communication all nodes only know that the k reporters are in $R^{\ell+1}$, where $|R^{\ell+1}| \geq \lfloor |R^\ell|/(c+1)^2 \rfloor$.*

PROOF. Consider the actions of the nodes when executing round $\ell+1$ of algorithm $\mathcal{A}$. Let $R^\ell_{(i,j)}$ denote the set of nodes that would send on channel $i \in \{0, 1, \ldots, c\}$ if they were a *reporter* and send on channel $j \in \{0, 1, \ldots, c\}$ if they were a *non-reporter*, where sending on channel 0 simply means that the node remains silent. Thus, there are exactly $(c+1)^2$ possible actions. Consequently, there must be a set $R^\ell_{(i',j')} \subseteq R^\ell$ of size $|R^\ell_{(i',j')}| \geq \lfloor |R^\ell|/(c+1)^2 \rfloor \geq k+2$. We now argue that we can set $R^{\ell+1} := R^{\ell}_{(i',j')}$ and that all nodes do not know anything about the reporters except that they are in $R^{\ell+1}$. Note that by definition, the k reporters all send on channel i' and the 2 or more non-reporters send on channel j', causing collisions on these channels. Of course, it is possible that $i' = j'$, in which case they only cause one collision, or even no collision if $i' = j' = 0$. Either way, no message is transmitted, and all nodes may at best learn that the reporters are in $R^{\ell}_{(i',j')} \cup R^{\ell}_{(j',i')} \supseteq R^{\ell}_{(i',j')}$. If the nodes in $V \setminus R^{\ell}_{(i',j')}$ cause additional collisions, the set of potential reporters may only become larger. It is possible that some of these nodes successfully transmit messages. However, since they do not possess any information that the other nodes do not already know from always listening on all channels, these transmissions cannot reduce the size of the set of potential reporters, which proves the claim. □

While Lemma 4 is used in the proof of Theorem 5 for small k, the next lemma strengthens our bound for large k .

Lemma 5. *Any deterministic information exchange algorithm with time complexity $\mathcal{O}(k)$ needs $\Omega(\log_k n)$ channels.*

Proof. A time complexity lower bound of $\Omega(k \log_k n)$ has been proven in the same model for one channel [13]. Any algorithm that is restricted to using one channel can simulate an algorithm that uses c channels by splitting each round into c sub-rounds and sending the messages that would be transmitted on channel $i \in \{1, \ldots, c\}$ in the i^{th} sub-round. Thus, simulating an algorithm on up to c channels takes at most c times longer, implying a time complexity of at least $\Omega(k\frac{\log_k n}{c})$ for any algorithm on up to c channels. Restricting the time to $\mathcal{O}(k)$ communication rounds, implies that the number of channels must be at least $\Omega(\log_k n)$. □

We can now prove the following theorem.

Theorem 5. *If $1 < k < n^{1-\varepsilon}$ for some constant $\varepsilon > 0$ and n is sufficiently large, any deterministic information exchange algorithm whose time complexity is $\mathcal{O}(k)$ needs $\Omega(n^{\Omega(1/k)} + \log_k n)$ channels.*

Proof. Lemma 4 states that there is an assignment of reporters such that the set of potential reporters shrinks at most by a factor of $(c+1)^2$ in each communication round, which entails that the set of potential reporters is larger than k after $\ell < \frac{1}{2}\log_{(c+1)}\left(\frac{n}{k}\right)$ rounds. Thus, the time complexity is at least $\Omega(\log_c \frac{n}{k})$. In order to achieve an upper bound of $\mathcal{O}(k)$, it must therefore hold that $c \in \Omega((n/k)^{\Omega(1/k)}) = \Omega(n^{\Omega(1/k)})$, where we use that $k < n^{1-\varepsilon}$. Hence the number of channels is at least $\Omega(\log_k n)$ according to Lemma 5. □

5. REFERENCES

[1] A. F. Anta, M. A. Mosteiro, and J. R. Muñoz. Unbounded Contention Resolution in Multiple-access Channels. In *Proc. 25th International Symposium on Distributed Computing (DISC)*, pages 225–236, 2011.

[2] H. Attiya, A. Bar-Noy, D. Dolev, D. Peleg, and R. Reischuk. Renaming in an Asynchronous Environment. *Journal of the ACM (JACM)*, 37(3):524–548, 1990.

[3] M. Bienkowski, M. Klonowski, M. Korzeniowski, and D. R. Kowalski. Dynamic Sharing of a Multiple Access Channel. In *Proc. 27th International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 83–94, 2010.

[4] M. Capalbo, O. Reingold, S. Vadhan, and A. Wigderson. Randomness Conductors and Constant-degree Lossless Expanders. In *Proc. 34th Annual ACM Symposium on Theory of Computing (STOC)*, pages 659–668, 2002.

[5] A. Castañeda, S. Rajsbaum, and M. Raynal. The Renaming Problem in Shared Memory Systems: an Introduction. Technical report, INRIA, 2010.

[6] B. Chlebus and D. Kowalski. Asynchronous Exclusive Selection. In *Proc. 27th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pages 375–384, 2008.

[7] B. Chlebus, D. Kowalski, and T. Radzik. Many-to-Many Communication in Radio Networks. *Algorithmica*, 54(1):118–139, 2009.

[8] B. Chlebus, D. Kowalski, and M. Rokicki. Average-time Complexity of Gossiping in Radio Networks. In *Proc. 13th International Colloquium in Structural Information and Communication Complexity (SIROCCO)*, pages 253–267, 2006.

[9] J. Czyzowicz, L. Gasieniec, D. R. Kowalski, and A. Pelc. Consensus and Mutual Exclusion in a Multiple Access Channel. In *Proc. 23rd International Conference on Distributed Computing (DISC)*, pages 512–526, 2009.

[10] S. Dolev, S. Gilbert, R. Guerraoui, and C. Newport. Secure Communication Over Radio Channels. In *Proc. 27th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pages 105–114, 2008.

[11] S. Gilbert, R. Guerraoui, D. Kowalski, and C. Newport. Interference-resilient Information Exchange. In *Proc. 28th IEEE Conference on Computer Communications (INFOCOM)*, 2009.

[12] S. Gilbert and D. Kowalski. Trusted Computing for Fault-Prone Wireless Networks. In *Proc. 24th International Symposium on Distributed Computing (DISC)*, pages 359–373, 2010.

[13] A. Greenberg and S. Winograd. A Lower Bbound on the Time Needed in the Worst Case to Resolve Conflicts Deterministically in Multiple Access Channels. *Journal of the ACM (JACM)*, 32(3):589–596, 1985.

[14] S. Holzer, Y. Pignolet, J. Smula, and R. Wattenhofer. Time-Optimal Information Exchange on Multiple Channels. In *Proc. 7th ACM SIGACT-SIGMOBILE International Workshop on Foundations of Mobile Computing (FOMC)*, 2011.

[15] D. Kowalski. On Selection Problem in Radio Networks. In *Proc 24th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pages 158–166, 2005.

[16] E. Kushilevitz and Y. Mansour. An $\Omega(D\log(N/D))$ Lower Bound for Broadcast in Radio Networks. *SIAM Journal on Computing (SICOMP)*, 27(3):702–712, 1998.

[17] K. Nakano and S. Olariu. Randomized Initialization Protocols for Ad Hoc Networks. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 11(7):749–759, 2000.

[18] D. E. Willard. Log-logarithmic Selection Resolution Protocols in a Multiple Access Channel. *SIAM Journal on Computing (SICOMP)*, 15(2):468–477, 1986.